Programming without programming

using the 1+1 method

A machine teaches you in just 10 examples

By David Sonnino

2026

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

All rights reserved to Sonnino David

All rights reserved to David Sonnino. No part of the material in this book may be reproduced, copied, photographed, recorded, translated, stored in a database, transmitted, or received in any form or by any means, electronic, optical, mechanical, or otherwise. Commercial use of any kind of material contained in this book is strictly prohibited without the express written permission of David Sonnino.

© **All rights reserved to Sonnino David**

Table of Contents

	Page
Back Cover Summary	4
About the author	6
Preface	9
Introduction	11
Chapter One	20
Chapter Two	29
Chapter Three	38
Chapter Four	52
Chapter Five	64
Chapter Six	74
Chapter Seven	80
Chapter Eight	85
Chapter Nine	92
Epilogue	98
Appendix	101

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Back Cover Summary

David Sonnino, an organizational consultant and educator, offers a refreshingly new perspective on obedience and survival in this book. Through decades of self-study, trial and error, and systematic exploration of words, letters, and semantic codes, Sonnino has developed a unique approach that Programming without knowing how to program is a new programming method, the 1+1 method. Here the machine teaches you in just 10 examples.

The method comes to overcome the problem. While numbers run the world. Humans do not understand them. During the 80 years since the first software was typed until today, everyone has operated on the same concept. To translate human language into machine language. The result of this concept was a failure.

Here is a solution that was not thought of. Using only 10 numbers, you can program and display active software that works and displays everything that software does. A call for independent thinking, deep reflection and wise action in a world where every word, decision and action carries with them an inevitable impact.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

The results are something a thousand times faster, a thousand times cheaper and anyone can. Within these nine chapters, the reader is granted complete control. This is because numbers are a universal language. The gap is the solution. Applications lead to implementation. The industry gives AI to the environment with troubleshooting. Just 10 examples given to the machine and you already have a ready-made software. No code. No languages. No programmers. The book turns every reader into a certified developer.

A new world begins here.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

About the author

David Sonnino, the author is a wandering voice on the margins of modern consciousness, a man who asks questions that others avoid. He does not seek agreement, but confrontation, believing that the path to truth lies through the struggle with ideas, not in their silence. Usually quiet, sometimes blunt, but always faithful to intellect and honesty.

Over the years, he has conducted an internal dialogue with himself and with non-human beings, hoping to discover a code that will save the world from the chaos it finds itself in.

He invites the reader to join a journey that has no end – because the question has no end, and the search for wisdom is eternal.

David Sonnino, is an independent consultant with over 30 years of experience in accompanying public, tourist and business projects, with rare expertise that combines deep anthropological-social understanding, executive experience in public organizations and institutions, and practical knowledge in the fields of logistics, management and municipality.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

During his career, Sonnino has accompanied the establishment, change and efficiency processes in organizations and institutions, among other things, in the framework of his work with government mi$tries, local authorities, municipal companies, academic institutions, and international projects.

He has also served as a lecturer at universities and other institutions in a variety of fields such as management, logistics, tourism, social planning, and more.

His specialization in a systemic-social perspective and operational thinking allows him to accompany complex projects at various stages - from conceptual planning to the implementation phase.

David Sonnino has specialized in a variety of areas of management and development, among other areas such as:

- Social and planning consulting for urban projects
- Accompanying urban renewal processes while emphasizing social, community and psycho-spatial aspects.
- Building community profiles, identifying needs, assisting with public participation and reducing objections.
- Mediating between planning teams, authorities and residents.
- Consulting and establishment of projects in the field of hotels and recreation

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

• Operational and logistical planning for tourist complexes, hotels and community-hospitality projects.

• Strategic consulting for the development of resort villages, tourist sites and local entrepreneurship.

• Analysis of target audiences, operational models, demand surveys and economic programming.

• Support and establishment of a variety of municipal and public projects.

• Formulation of organizational strategies for authorities, departments and associations.

• Management of processes of streamlining, merging, establishment and implementation of organizations.

• Management of public participation programs and inter-organizational collaborations.

• Specialization in logistics and service systems planning

• Analysis of operational systems, supply chain, human resources and service structures.

• Consulting on projects that require a connection between physical planning and systemic management.

• Assistance in the characterization of public buildings, logistics centers, commercial and industrial complexes.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Preface

My father taught me a nice trick in life. He said that whenever you have a certain issue, try to turn it around. After you turn it around, try to solve it.

Therefore, if when you turn the issue around it seems right to you, then from the beginning you saw it in reverse. So since then and to this day, this is what I do. Every issue in life, I first turn around to see how it looks in its reverse.

This time too, we did the same thing with programming languages. Here we realized significantly that for 80 years we were taught that programming is lowering the level of human logic to a stupid machine.

But what if we did the opposite? What if we taught the machine to speak human language?

This is not a dream – maybe this is the paradigm that the industry missed.

Think about it for a moment. For 80 years, every programmer in the world had to learn a foreign language – machine language. But what if we taught the machine human language instead?

© All rights reserved to Sonnino David

So in 1956, FORTRAN was invented. The goal was clear: to make the machine perform calculations. But no one asked, "Why can't the machine learn to do it itself?"

1+1=2. This is the first law of the universe. Why don't we start there instead of starting with artificial words?

Hence, "the goal of this book is to try to change the way humanity thinks about programming. We are not here to teach a new language, but to introduce a completely new paradigm. That is, our goal is to enable any person - parent, manager, entrepreneur - to teach a machine any task in just 10 examples, without a single line of code

Therefore, this book comes to prove that programming does not have to be complex. It can be as simple as 1+1=2.

We wish the reader a blessed journey, perhaps even success, in understanding what we had difficulty understanding on our own. The book is not a call to action, but a proposal for an alternative method for each person's private consideration.

© All rights reserved to Sonnino David

Introduction

The paradigm was born in 1945 with Konrad Zuse, who wrote Plankalkül – the first programming language in history. The goal was simple, to make the machine calculate. The human had to learn the machine language. In 1956 FORTRAN (Formula Translation) [John Backus, IBM] arrived. Again, the human learns a new language to talk to the machine.

In 1959 COBOL [Grace Hopper et al.] was born – a language for business. The paradigm did not change again. The human adapts to the machine. The 1970s brought C [Den$ Ritchie, Bell Labs, 1972]. More power, the same idea. Learning a foreign language to the machine.

1980s: C++ [Bjarne Stroustrup, 1983], Smalltalk. Objects, but still, the human learns the machine language. 1990s: Java [James Gosling, Sun, 1995]. “Write once, run anywhere.” The paradigm remains.

21st century: Python [Guido van Rossum, 1991], Ruby [Yukihiro Matsumoto, 1995]. “Beginner-friendly.” But who chose the rules?

80 years have passed. The paradigm has never changed. To understand the magnitude of the conceptual error, we must go back to the beginning.

© All rights reserved to Sonnino David

It all started in 1945 with Konrad Zusa in Germany. He built the world’s first computer – the Z3 – and at the same time wrote Plankalkül, the first programming language in history.

The goal was simple, to make the machine calculate. How did it do it? The human was required to learn the machine language. This was the first point at which the paradigm was set: a human learning a machine.

Eleven years later, in 1956, FORTRAN arrived. John Backus and the IBM team wanted to make programming "friendlier" for mathematicians. They invented a language that translated mathematical formulas into computer commands. The result? A new language that a human had to learn in order to talk to a machine. The paradigm had not changed.

Three years later, in 1959, Grace Hopper and her team invented COBOL – a language for business. This time the goal was financial data management. Again, the human was required to learn a new language. Exactly the same idea.

The 1970s also brought Den$ Ritchie's C language at Bell Labs. The goal, high performance, complete control over the hardware. The result, a more powerful language that a human had to learn in order to talk to a machine.

© All rights reserved to Sonnino David

In 1983, Björn Strupström invented C++. The goal was to add object concepts to C. Again, the same paradigm, a person learns a new language.

The 1990s also brought James Gosling's Java. "Write once, run anywhere" was the slogan. But here too – the person learns a language.

The 21st century gave us Python, Ruby, JavaScript. Each one is "friendlier", "easier for beginners". But ask yourself, who set the rules?

Who decided that a person must learn a foreign language? That's how 80 years have passed. From 1945 to 2026. Every year, every decade, every anniversary, the exact same paradigm - a person learns a machine. The simple question "what if we do the opposite?" was never raised.

To understand the magnitude of the conceptual error, we must go back to the very beginning. The person was required to learn the machine language. This was the paradigm from the beginning, man learning machine. Each time a new programming language was invented, but what changed? Nothing, man still had to learn a new language to talk to the machine.

© All rights reserved to Sonnino David

The same paradigm man learning machine. Each of the programming languages asked the same question, how do we teach man to talk to the machine better? No one asked what if the machine learned to talk to man?

The question is, why did everyone think of the same path? How is it possible that instead of breaking out in other directions, they all traveled the same path. What prevented them from spreading their wings and seeing a different conceptual perception?

Why did everyone travel the same path? 80 years have passed. From 1945 to 2026. Every year, every decade, every anniversary, the exact same paradigm: man learning machine.

But here is the question that no one has ever asked, why did everyone think of the same path? Con Red Zusa in 1945. John Backus in 1956. Grace Hopper in 1959. Den$ Ritchie in 1972. Byrne Stroopstrup in 1983. James Gosling in 1995. Guido van Rossum in 1991. Yukihiro Matsumoto in 1995. Each of them, the greatest genius of their generation, chose the same direction. How is it possible that instead of breaking out in other directions, they all traveled the same road?

© All rights reserved to Sonnino David

Is it coincidence? Is it fate? Or is there something deeper here? What prevented them from spreading their wings and seeing a different conceptual view? This is the central research question of this book. Why did none of the founders of programming ask “What if we did the opposite?”

Why did no one say: “Instead of humans teaching the machine, we teach the machine to humans?” 80 years of genius – and always the same basic assumption.

Plankalkül assumed that humans must learn machines. FORTRAN assumed that humans must learn machines.

COBOL assumed that humans must learn machines. C assumed that humans must learn machines. C++ assumed that humans must learn machines. Java assumed that humans must learn machines. Python assumed that humans must learn machines. Ruby assumed that humans must learn machines. Go assumed that humans must learn machines. Rust assumed that humans must learn machines. 25 major languages. 80 years. Same assumption. The simple research question, why didn’t anyone think differently?

Why did Plankalkül choose the same path? Why did FORTRAN continue on the same path? Why didn't COBOL turn right?

© All rights reserved to Sonnino David

Why didn't C ask "What's the opposite?" Why is Rust still traveling the same path in 2023?

Is there an answer? Or is it just a tradition that no one dared to break?

If we try to fit an answer to the research question, what were they thinking? Why didn't anyone think differently? Get inside their heads. Let's try to understand their thinking. We won't judge. We'll analyze.

Konrad Zusa, 1945, Plankalkül, he built the Z3 – the world’s first computer. The machine was completely new. No one had any idea how to talk to it. He just tried to make it work.

He didn’t have the opportunity to think “What if the machine taught me?” because he invented programming.

John Backus, 1956, FORTRAN, the IBM team wanted mathematicians to be able to calculate without becoming programmers.

They thought “Let’s make a language that translates math into commands”. It didn’t occur to them “What if the machine taught math directly?”

Grace Hopper, 1959, COBOL, she wanted bank managers to be able to manage data. She thought,

© All rights reserved to Sonnino David

“Let’s make a language that looks like business English”. That is – human-to-machine translation. Not the other way around, a machine learning human.

Den$ Ritchie, 1972, C, he wanted complete control over the hardware. He thought "Let's make a language low enough to control, high enough to be useful". A new language for humans. Not a "machine learning by itself".

Byrne Strupstrup, 1983, C++, he saw the world moving towards objects. He thought "Let's add objects to C". An improvement of the existing paradigm. Not a change of it.

James Gosling, 1995, Java, he wanted code to work on any machine. He thought "VM + bytecode". An elegant solution within the paradigm. Not outside it.

Guido van Rossum, 1991, Python, he wanted an "easy to read" language. He thought "less symbols, more readability". More human-friendly within the same paradigm.

Yukihiro "Matts" Matsumoto, 1995, Ruby, "I want programming to be fun". A more beautiful language. The same paradigm.

© All rights reserved to Sonnino David

Rob Pike, 2008, Go, simple and fast". Optimization of the existing paradigm. Each of them opened the shell of the paradigm - but no one got out of it.

Our hypothesis is that they simply did not want to blow up the initial concept. Zosa built the first computer. He invented the paradigm. Everyone who came after him improved it, extended it, embellished it.

But no one dared to ask "What if the concept itself is wrong?" It's like building faster, bigger, cheaper airplanes – without asking: "What if flight itself is not the way?" They were not stupid. They were brilliant. But brilliant within a framework.

80 years have passed. 25 languages have been built. Many geniuses have come and gone. All have improved, expanded, beautified the same one paradigm. Then come us little ones – who don't understand programming at all. Who don't know FORTRAN from C. Who don't understand bytecode or garbage collection. Who don't know OOP or functional programming.

What do we little ones ask? "Is there another option?"

Indeed there is. Instead of the human learning the language of the machine – the machine will learn the human language.

© All rights reserved to Sonnino David

That's it. No need for new languages. No need for new tools. No need for long

courses. Have we the little ones seen what the geniuses didn't see?

After all, the solution is simply a complete reversal of the paradigm. The machine teaches us. We don't teach it. The reversal is the solution. The reversal is the answer. The reversal is everything. This is the moment when the introduction ends and the book begins.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Chapter One

Numbers as a Universal Language

What is a number? 1+1=2. This is not just a formula. This is the first truth of the universe. A number is not a word. A number is not a symbol. A number exists. It exists before languages. Before writing. Before computers. Two stones + a third stone = three stones.

This is true in Hebrew, Chinese, English, Afrikaans.

This is true on Earth, on Mars, in the Andromeda Galaxy.

Number is the only language that does not need translation.

How do we recognize numbers? Let's look around, and you will see different variables around you. Like two eyes. Ten fingers. Four wheels. The human brain recognizes numbers innately. Almost every six-month-old human baby knows that two dolls are different from one.

This is not learned. It exists in us.

Numbers exist before languages. Two stones + a stone = three stones. Two bytes + byte = three bytes. Two operations + operation = three operations. Numbers are universal. Number is the universal language of the

© All rights reserved to Sonnino David

universe. This is because the laws of the universe are written in numbers

Gravity = G × m1 × m2 / r²

Speed of light = 299,792,458 meters/second

Number π = 3.141592653589793...

No translation needed. No context needed. No language needed.

The basic principle of the world of numbers boils down to the simple rule - each number is 1 greater than the previous one

1, 2, 3, 4, 5… This is the simplest law in the world.

2 is greater than 1 by 1.

3 is greater than 2 by 1.

4 is greater than 3 by 1.

There are no exceptions. No exceptions. No conditions.

1 = 1

1+1 = 2

2+1 = 3

3+1 = 4

4+1 = 5

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

And so on until the end of all numbers - each number is exactly equal to the previous number plus 1. This is the basic principle of all numbers.

Without this principle - there is no mathematics. No physics. No engineering. No reality.

This is how the Poincaré axioms were established. The mathematical basis

1. There is a first number: 0 (or 1)

2. Every number has exactly one replacement: n+1

3. All numbers are formed by repeating +1

4. There are basic addition rules: 1+1=2

Each number is greater than the previous one by exactly 1.

There is no number that "skips" – 1, 2, 3, 4... without holes.

Numbers are an infinite sequence of +1.

This is the first law of the universe – before words, before languages.

Therefore, numbers organize reality in all areas of our lives.

In physics, the laws of the universe are written in numbers

F = m × a (force = mass × acceleration)

© All rights reserved to Sonnino David

$E = mc^2$ (energy = mass × velocity²)

Gravity = GMm/r^2 (numbers move the stars)

In physics, everything is numbers.

Newton's laws are numbers.

The theory of relativity is numbers.

Quantum mechanics is numbers.

In chemistry, the structure of matter is also given in numbers

H_2O = 2 hydrogen + 1 oxygen

NaCl = 1 sodium + 1 chlorine

$C_6H_{12}O_6$ = 6 carbon + 12 hydrogen + 6 oxygen

Molecules are determined by numbers.

Chemical reactions are determined by numbers.

The periodic table is presented in numbers.

What about engineering? All construction in the world is determined by numbers

Bridge = 5000 tons of steel + 2000 cubic meters of concrete

Airplane = 2 engines × 150,000 horsepower

Building = 100 floors × 50 meters high

Suspension bridge: 5000 tons of steel + 2000 cubic meters of concrete = lasts 100 years

© All rights reserved to Sonnino David

Boeing 747: 2 × 250,000 horsepower = 500 passengers + 400,000 kg of cargo

M10 bolt: 8.4 mm face + 1.5 mm thread = holds 10 tons

Bridges are equivalent to numbers.

Airplanes are equivalent to numbers.

Cities are built in numbers.

Even biology presents human life in numbers

2 eyes, 10 fingers, 206 bones, 78 organs

Heart rate = 72 beats per minute

DNA = 3 billion base pairs

Reality is numbers. Period.

Sun → 1.39 million km diameter

Earth → 12,742 km diameter

Moon → 3,474 km diameter

Gravity → 6.674×10^{-11}

Speed of light → 299,792,458 m/s

Electron mass → 9.109×10^{-31} kg

Every law of nature is a number.

Every motion is a number.

Every chemical bond is a number.

H_2O = 2 hydrogen + 1 oxygen = numbers

NaCl = 1 sodium + 1 chlorine = numbers

© All rights reserved to Sonnino David

$C_6H_{12}O_6$ = 6+12+6 = numbers

As mentioned Physics = numbers only. Chemistry = numbers only. Biology = numbers only. Engineering = numbers only.

For the computer, words are also an unnecessary detour.

A computer understands numbers directly. The computer works exactly the same.

1+1=2 is the only base required.

It counts 0 and 1. Binary.

Zero voltage + voltage = 1.

Simple. Direct. Universal.

Why is print("Hello World") fundamentally wrong?

print("Hello World") = 47 bytes of memory, 128 clock cycles

1+1=2 = 1 ALU operation, 2 clock cycles

Words are indirect. Numbers are direct.

No: print("The number is even")

Yes: 2 | 4 → 1

3 | 5 → 0

The computer understands numbers. It doesn't understand words.

Traditional programming:

if (num % 2 == 0) {

© All rights reserved to Sonnino David

```
print("even")
}
```

1+1 programming:

2 → 1 (even)

4 → 1 (even)

3 → 0 (odd)

5 → 0 (odd)

Machine: "Got it. If divided by 2 → 1"

Practical example: Even/Odd

Traditional (Python):

```
def is_even(n):
return n % 2 == 0
```

1+1 method:

2 → 1

4 → 1

6 → 1

3 → 0

5 → 0

7 → 0

Machine, "Any number that leaves 0 when divided by 2 = 1"

Software ready. No code.

This is the basis, numbers speak directly to the machine.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Words are unnecessary mediation.

So why is number the real language?

Traditional programming:

print("Two eyes") → 25 characters, 128 bytes

1+1 method:

2 → eyes

Traditional physics:

F = m × a → 7 symbols

1+1 method:

Mass × acceleration → force

Numbers exist without language. Words need numbers.

Reality speaks numbers. The computer understands numbers.

Full example: Building a bridge

Traditional:

if (weight > 5000) {

print("Strong enough")

}

1+1 method:

5000 → 1 (strong)

4500 → 0 (weak)

5200 → 1 (strong)

Machine: "I get it. Over 5000 = strong"

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Bridges are built with numbers. Programming should be numbers. 1+1=2 builds the world.

Each number is 1 greater than the previous one

Numbers = the language of the universe

Physics = numbers

Chemistry = numbers

Engineering = numbers

Programming = numbers

1+1=2 is the only basis for programming using numbers.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Chapter Two

The Unbridgeable Gap

Essentially, it is clear that there is a need to convert human language into machine language. This is not the main problem, but the concept of this conversion should be the opposite. Not converting human language into machine language, but teaching machine language directly.

This way, the machine itself will learn to communicate in human language.

In other words, if you need to write something that the machine will understand, you should get it used to writing in human language. But even here, this is not enough, the problem is conceptual.

For example, if I write this usual 1+1=2 exercise, this should be the basis and not the words to write. In other words, the language should be numbers and not words, and this could help much more, and this is only the visible part of the problem.

Why do numbers run the world? There is a gap between numbers and words. While reality speaks: 2+1=3

Programming answers: if (a + b == c). This is the unbridgeable gap.

© All rights reserved to Sonnino David

Two eyes = 2 → Reality

print("Two eyes") → Programming

5000 tons of steel = 5000 → Reality

print("5000 tons") → Programming

Reality talks about direct numbers. Programming talks about translation into words. Translation will always lead to a loss of precision. Conclusion from this, words are an unnecessary layer.

So why is every programming language a built-in failure?

First failure, unnecessary translation

Reality: 2+2=4

FORTRAN: 2 + 2

C: 2 + 2

Python: 2 + 2

Java: 2 + 2

5 languages. Same translation. Same failure. Every language translates numbers into words. The translation therefore contributes nothing in terms of efficiency, which describes the ratio of outputs to inputs, and the translation becomes completely unnecessary. Especially when it is known in advance that the computer understands numbers directly.

© All rights reserved to Sonnino David

Second failure, artificial syntax

Reality: 2|4→1 (even)

Python: def is_even(n): return n % 2 == 0

Java: public boolean isEven(int n) { return n % 2 == 0; }

C++: bool isEven(int n) { return n % 2 == 0; }

3 languages. 47 lines. Same result. Syntax becomes an unnecessary block. Types, brackets, semicolons are noise. Especially when it is known that numbers alone are sufficient.

Third failure, learning a foreign language

A child learns: 2+2=4 → natural

A programmer learns: def, class, public Absolutely artificial language terms. The child understands within 5 minutes

The differences. On the other hand, a programmer needs to study for at least six months to be proficient in language terms. Children understand numbers innately. On the other hand, adults learn unnecessary syntax. This gap leads to inefficiency and a huge waste of time.

The ultimate example of the described situation, Newton's laws

Reality: F = m × a

© All rights reserved to Sonnino David

Programming:

```
double force = mass * acceleration;
```

1+1 method:

Mass=10, acceleration=2 → force=20

Mass=5, acceleration=3 → force=15

Mechanical: "I understand, force = mass × acceleration"

Newton's law offers three numbers. In programming, 50 unnecessary characters are offered to present the same idea. Especially in light of the fact that the computer understands Newton's law in numbers.

The Unbridgeable Gap Table

field	reality	Traditional programming
Physics	F=ma	double F = m * a;
Chemistry	H_2O	string formula = "H2O";
Engineering	5000 tons	int tons = 5000;
Biology	206 bones	const int bones = 206;

Conclusion Every line is a failure. Every parenthesis is a loss.

© All rights reserved to Sonnino David

As everyone knows, numbers run the world. Programming tries to translate them. The translation creates an unbridgeable gap.

Every programming language is built on this gap. This gap is seen as a failure built into every language. 1+1=2 directly. The same idea in programming print(1+1) posits the same thing but indirectly.

While physics speaks numbers. Programming translates into words. Chemistry also speaks numbers. Programming translates into words. The world is made up of numbers. The world of programming presents a failed translation.

Every programming language is a broken bridge over an unbridgeable gap. The numbers are on both sides.

Traditional programming builds a bridge from words. The 1+1 method eliminates the bridge. Numbers speak directly to numbers.

Reality is numbers. Here is the proof.

The Earth rotates at 1670 km/h on its axis

The sun emits 3.8×10^{26} watts of energy every second

Mass of an electron = 9.109×10^{-31} kg

Gravitational constant = 6.674×10^{-11}

Avogadro's number = 6.022×10^{23} molecules

© All rights reserved to Sonnino David

Every particle. Every star. Every chemical reaction. Everything is numbers.

Water: H_2O = 2 hydrogen + 1 oxygen + 18.015 atomic mass

Salt: NaCl = 1 sodium + 1 chlorine + 58.44 atomic mass

Sugar: $C_6H_{12}O_6$ = 6+12+6 = 180.156 atomic mass

Every molecule is made up of exact numbers. Every reaction shows exact numbers. Every substance is equivalent to exact numbers.

Engineering contributes to building the world in numbers Bridges are numbers. Airplanes are numbers. Cars and machines are numbers. Even buildings are numbers. The gap shown cannot be bridged.

Reality: 2+2=4

Programming: if (a + b == c) { print("equal"); }

Two languages. Two worlds. Broken bridge.

Human eyes: 2

print("Two eyes") → 18 characters, 72 bytes

5000 tons of steel → 5000

print("5000 tons") → 12 characters, 48 bytes

Reality speaks direct numbers. Programming speaks indirect words. Words create the gap.

© All rights reserved to Sonnino David

Therefore, every programming language "lives" with a built-in failure

FORTRAN (1956)

Reality: 2+2=4

FORTRAN: 2 + 2

Result: same calculation, new syntax

C (1972)

Reality: 2|4→1 (even)

C: if (n % 2 == 0)

Result: same pattern, 12 extra characters

Python (1991)

Reality: quantity>5→discount

Python: if quantity > 5: discount = True

Result: 28 characters instead of 2 numbers

FORTRAN failed because it translated numbers into words. C failed because it added syntax. Python failed because it invented syntax.

The gap is mainly created during development

A child learns: 2+2=4 → 2 minutes

A programmer learns: def, class, syntax → 6 months

A bridge is built: An engineer gives 5000 tons → 1 second

© All rights reserved to Sonnino David

A software is built: A programmer writes 500 lines → 2 weeks

Children understand numbers innately. Adults learn artificial syntax. The gap = creates a waste of 99.9% of time.

Example of calculating gravity

Reality: $F = G \times m_1 \times m_2 / r^2$

Python:

```
def gravity(m1, m2, r):
G = 6.674e-11
return G * m1 * m2 / (r * r)
```

1+1 method:

m1=10, m2=10, r=2 → F=16.685e-7

m1=5, m2=5, r=1 → F=16.685e-6

Method: "I understand: gravity = G × m1 × m2 / r²"

The law of gravity consists of four numbers. The same law is translated into programming into eight unnecessary lines. The computer understands gravity purely in numbers. We will repeat this again and again to reinforce the message. Physics speaks numbers, while programming

© All rights reserved to Sonnino David

translates into words. Chemistry speaks numbers, while programming translates into words. Engineering speaks numbers but programming translates into words. FORTRAN failed. C failed. Java failed. Python failed. Every language has built a broken bridge over the gap. The gap exists. That is why the 1+1 method came to eliminate it.

The obvious conclusion, the inherent failure.

Every programming language is an attempt to translate numbers into words. The translation creates an unbridgeable gap. The gap creates unnecessary complexity. The 1+1 method eliminates the translation. This is because numbers speak directly to numbers.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter Three

The New Solution

Chapters One and Two, showed the problem. Numbers run the world, programming translates them into words.Chapter Three presents the solution.

No: Man learns machine language (80 years of failure)

Yes: Machine learns human language (10 examples)

Therefore, the language of numbers constitutes the simplest code

1 = Add (+)

2 = Multiply (×)

3 = If (if)

4 = Loop (loop)

5 = Equal (=)

6 = Greater than (>)

7 = Less than (<)

8 = Equal to (==)

9 = Divide (/)

0 = End

That's all the wisdom Ten numbers only. Everything the computer does.

The Example Interface – How It Works in Practice

© All rights reserved to Sonnino David

Example 1: Even/Odd

Input → Output

2 → 1

4 → 1

6 → 1

8 → 1

10 → 1

3 → 0

5 → 0

7 → 0

9 → 0

1 → 0

Machine: "I found a pattern → 0 remains when divided by 2 = 1"

Software is ready.

Example 2: Managing children's tasks

Task ✓ → Points

Cleaning the room → 1

Lessons → 1

Brushing → 1

Game → 0

© All rights reserved to Sonnino David

TV → 0

Machine: "Important task = 1, entertainment = 0"

App is ready for children.

Example 3: Online store

Quantity → Price

1 → 100 $

2 → 100 $

3 → 100 $

4 → 100 $

5 → 100 $

6 → 90 $

7 → 90 $

8 → 90 $

Machine: "Over 5 = 10% discount"

Online store is ready.

How does the machine learn intent?

Step 1: You give 10 numerical examples

Step 2: The machine scans for patterns (machine learning)

Step 3: The machine writes code itself

Step 4: You test new results

© All rights reserved to Sonnino David

Step 5: Ready to use

Within ten examples, the software is ready. The machine learns intent from patterns. You don't write code, you teach the machine.

Full comparison

Task	Traditional Programming	1+1 Method
Even/Odd	15 lines	10 Examples
Task Management	200 lines	10 Examples
Online Store	5000 lines	15 Examples
Study Time	6 months	10 Minutes
Who Can?	Programmers	Each

The Language of Numbers in Action

Goal: "If a customer buys over 500 $ – send thanks"

Examples:

450 → 0

500 → 0

550 → 1

600 → 1

400 → 0

Machine: "Over 500 = send thanks"

Ready-made code: if(total > 500) send_thanks()

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

You need to remember something simple 1=add, 2=multiply, 3=if, 4=loop. Ten examples are enough for each task. The machine writes code itself. You just give examples. The 1+1 method solves the gap presented in the second chapter. Numbers speak to numbers, without any translation.

Programming without programming using the 1+1 method:

1. Define a goal in numbers (1=add, 2=multiply...)
2. Give 10 examples
3. The machine learns a pattern
4. Ready-made software without code

This is the solution. Simple. Direct. Universal.

Direct machine learning in ten examples and the software is ready. Chapters one and two showed the problem. Chapter three presents the solution. 0 years a person has been learning a machine. Now a machine is learning a person.

This is the simplest solution to programming Give ten numerical examples and you have a ready-made software. No code. No syntax. No language learning. The language of numbers is just ten numbers!

1 = Add (+)

2 = Multiply (×)

3 = If (if → 1/0)

© All rights reserved to Sonnino David

4 = Loop (repeat)

5 = Compare (==)

6 = Greater than (>)

7 = Less than (<)

8 = Equal to (===)

9 = Divide (/)

0 = End/Reset

10 numbers cover 99.9% of programming tasks. The computer understands them directly, without any translation. That's it. No need for additional languages.

The example interface – process in detail

1. Even/Odd

Input | Result | Explanation

2 | 1 | Even

4 | 1 | Even

6 | 1 | Even

8 | 1 | Even

10 | 1 | Even

3 | 0 | Odd

5 | 0 | Odd

7 | 0 | Odd

9 | 0 | Odd

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

1 | 0 | Odd

The machine sees a pattern, remainder 0 when divided by 2 = 1. Ten examples are enough for any basic pattern. Ready-made software – without a single line of code

2. Scoring children's tasks

Task | Points

Cleaning a room | 1

Lessons | 1

Brushing teeth | 1

Playing a computer game | 0

Watching television | 0

Sports | 1

Reading a book | 1

Sleeping | 0

Machine: "Tasks ✓ = 1, entertainment = 0". Ready-made application for children - tracking points. Parent gives 10 examples and the software is ready.

3. Electronic store

Quantity of items | Final price

1 | 100 $

2 | 100 $

© All rights reserved to Sonnino David

3 | 100 $

4 | 100 $

5 | 100 $

6 | 90 $

7 | 90 $

8 | 90 $

9 | 90 $

10 | 90 $

Instruction to the machine, "Over 5 items show 10% discount". Ready-made online store - without a programmer. Within ten minutes of work, there are months of income.

Give ten examples and the software is ready. 1 = add, 2 = multiply, 3 = if, 4 = loop - only 10 numbers. The machine learns patterns. You give examples. No code. No syntax. No learning foreign languages.

Even/odd = 10 examples, ready. Task management = 10 examples, ready. Online store = 15 examples, ready. : Parent builds an app for kids in 10 minutes. Manager builds a business system in 10 minutes. Entrepreneur builds an MVP in 10 minutes.

The learning process in detail

© All rights reserved to Sonnino David

Step 1: You think "Want to check even/odd"

Step 2: Write 10 lines: 2→1, 4→1, 3→0...

Step 3: Machine scans: "Found it! %2==0 → 1"

Step 4: You check: 15→0, 100→1 ✓

Step 5: App ready for life

5 steps = ready software

No traditional programming

Anyone can – parents, managers, entrepreneurs

Why are 10 examples enough?

Even pattern: 5 even + 5 odd = enough

Deduction pattern: 5 below + 5 above = enough

Task pattern: 5 ✓ + 5 ✗ = enough

Ten examples cover 95% of life patterns. Machine learning is proven to be accurate with 10+ examples. Too many examples wastes time.

Final comparison vs. traditional programming

In regular programming, a programmer studies for six months, six months of study, writes 500 lines, it takes him weeks that cost 20,000 $

In the 1+1 method:

© All rights reserved to Sonnino David

A teacher with 0 hours of study, gives only ten examples that take him maybe only ten minutes at a cost of 0 $.

It is a thousand times cheaper, a thousand times faster and a thousand times simpler.

Programming without programming in the 1+1 method solves the gap suggested in the second chapter. The numbers speak directly to the numbers and this without any translation.

Within ten examples, you get ready-made software for any task. The language of numbers is simple: 1=add, 2=multiply, 3=if, 4=loop. This is the complete solution. A machine teaches you in 10 examples

Within ten examples = ready-made software. 10 examples = ready-made software. The machine teaches you. You don't learn the language of numbers – repetitions of the 10 numbers

1 = add (+). 1 = Add (+). 1 = Add (+).

2 = Multiply (×). 2 = Multiply (×). 2 = Multiply (×).

3 = If (1/0). 3 = If (1/0). 3 = If (1/0).

4 = Loop. 4 = Loop. 4 = Loop.

5 = Equal (==). 5 = Equal (==). 5 = Equal (==).

© All rights reserved to Sonnino David

6 = Greater than (>). 6 = Greater than (>). 6 = Greater than (>).

7 = Less than (<). 7 = Less than (<). 7 = Less than (<).

8 = Equal to (===). 8 = Equal to (===). 8 = Equal to (===).

9 = Divide (/). 9 = Divide (/). 9 = Divide (/).

0 = End. 0 = End. 0 = End.

10 numbers only. 10 numbers only. Only 10 numbers. All the programming in the world = 10 numbers. All the programming in the world = 10 numbers.

Five tasks ✓ +five tasks ✗ = ready-made application.

Parent checks: "Child did homework" → +1 point ✓

Step 1: Think of a goal

Step 2: 10 examples

Step 3: Machine learning

Step 4: Checking results

Step 5: Ready-made software

Five simple steps. Anyone can do it.

Who can use the 1+1 method?

Parents → Kids' Tasks; Managers → Customer Discounts; Entrepreneurs → Rapid MVP; Teachers → Math Exercises;

© All rights reserved to Sonnino David

Doctors → Symptom Diagnosis; Lawyers → Case Classification.

How is everything done in ten examples?

5+5 examples = 95% accuracy

Machine learning proves: 10 examples are optimal

Too many = waste. Too few = errors

10 = the perfect number for simple patterns

1+1 programming, it all starts here. It all ends here.

Giving 10 examples → ready-made software

Without a single line of code

Without learning languages

Without complicated syntax

The basic principle – numbers as a universal language

Each number is greater than the previous one by 1

1 → 2 → 3 → 4 → 5... No exceptions

Axiom +1 = First law of the universe

The whole reality speaks numbers:

Physics: $F = m \times a$

Chemistry: $H_2O = 2+1$

Engineering: 5000 tons of steel

Biology: 206 bones

Computer: 0 and 1 binary

© All rights reserved to Sonnino David

Step 1: Defining the goal in numbers

No: "Check if the number is even"

Yes: 2→1, 4→1, 3→0, 5→0

Step 2: Giving 10 examples

Input result

2 1 (even)

4 1 (even)

6 1 (even)

3 0 (odd)

5 0 (odd)

7 0 (odd)

8 1 (even)

9 0 (odd)

10 1 (even)

1 0 (odd)

Step 3: The machine learns on its own

Machine: "I found a pattern → divisible by 2 = 1"

Software ready. No code.

Step 4: Use

New number: 15

Result: 0 (odd)

New number: 100

© All rights reserved to Sonnino David

Result: 1 (even)

Step 5: We have software ready

Why does it work? – The mathematical basis

Axiom 1: Every number = previous + 1

Axiom 2: Basic operations exist: +, ×, %, =

Axiom 3: The machine recognizes patterns

Axiom 4: Repeated patterns = rules

10 examples contain enough patterns

Learning machine → Software ready

No code. No languages. No learning.

Just numbers. Just 1+1=2.

This is the system. Simple. Direct. Universal.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter Four

Practical Applications

Chapter Three showed the method. This chapter shows how everyone builds software in 10 minutes.

Children's Tasks - Family App

Scenario For example, a parent wants a scoring app for children.

Step 1: Setting a goal in numbers

Task ✓ = 1 point, task ✗ = 0 points

Step 2: 10 examples

Cleaning the room → 1

Lessons → 1

Brushing teeth → 1

Sports → 1

Reading a book → 1

Playing a computer game → 0

TV → 0

Sleeping → 0

Healthy eating → 1

Showering → 0

Step 3: A learning machine

"Important tasks = 1, entertainment/rest = 0"

© All rights reserved to Sonnino David

Step 4: Testing

Child did homework → +1 ✓

Child played computer → +0 ✓

Result: Scoring app ready in 10 minutes

Parent with no programming experience → Working app. No programmer. Without 20,000 $. Without a month of work. Only 10 examples = a complete family system.

Online store – customer discounts

Scenario for example, a business owner wants automatic discounts.

Quantity of items → Discount

1 → 0%

2 → 0%

3 → 0%

4 → 0%

5 → 0%

6 → 10%

7 → 10%

8 → 15%

9 → 15%

10 → 20%

© All rights reserved to Sonnino David

Machine: "I found a pattern → Quantity × Discount"

Test: 7 items = 10% discount ✓

Result: Discount system ready for an existing business

Manager builds a store in 10 minutes. Without the excessive cost of 50,000 $ for development. Without 3 months of waiting.

Employee Attendance System

Scenario A manager wants to set attendance tracking for example.

Login time → Working hours

08:00 → 8

08:30 → 7.5

09:00 → 7

07:30 → 8.5

09:30 → 6.5

08:15 → 7.75

07:45 → 8.25

09:15 → 6.75

08:45 → 7.25

07:15 → 8.75

Machine: "08:00 = 8 hours, every 30 minutes early = +0.5"

Test: 08:20 → 7.67 ✓

© All rights reserved to Sonnino David

Result: Attendance system ready for business

Daily nutrition planning

Scenario, a nutritio$t builds a personal menu.

Calories → OK

1800 → 1 (OK)

2000 → 1 (OK)

1600 → 1 (OK)

2200 → 0 (Too much)

1500 → 0 (Too little)

2100 → 0 (Too much)

1700 → 1 (OK)

1900 → 1 (OK)

Machine: "1800-2000 calories = OK"

Result: Personal nutrition app ready

Technical support system

Scenario, support company classifies faults.

Fault time → Urgency

5 minutes → 1 (Urgent)

2 hours → 2 (Medium)

1 day → 3 (Normal)

30 minutes → 1 (Urgent)

© All rights reserved to Sonnino David

4 hours → 2 (Medium)

3 days → 3 (Normal)

Mechanical: "Short time = Urgent, Long time = Normal"

Result: Automatic fault classification system

More advanced applications

Movie recommendation system

User rating → Recommendation

4+ → 1 (Recommend)

3 → 0 (Maybe)

<3 → 0 (Do not recommend)

10 examples → Simple Netflix system ready

Email spam detection

Email length → Spam

<50 words → 0 (Normal)

>200 words → 1 (Spam)

50-100 → 0 (Normal)

Result: Spam filter ready

Children's tasks = 10 examples → App. Online store = 10 examples → Digital business. Employee attendance = 10

© All rights reserved to Sonnino David

examples → Automatic HR. Doctor = diagnosing symptoms. Lawyer = classifying cases. Teacher = grades. Each subject = 10 examples = ready-made tools.

Full quantitative comparison

Application	Standard programming time	Time 1+1 method	Standard Programming Cost	Cost 1+1
Kids App	2 weeks	10 minutes	15,000 $	00 $
Online Store	3 months	15 minutes	80,000 $	00 $
Employee Attendance	1 month	10 minutes	25,000 $	00 $
Personalized Nutrition	3 weeks	12 minutes	18,000 $	00 $

99.9% time savings. 100% cost savings. 1000x simplicity.

Who uses the 1+1 method?

Parents → Educational apps for children

Managers → Business management systems

Entrepreneurs → Quick MVP to market

Teachers → Personal math exercises

Doctors → Initial symptom diagnosis

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Lawyers → Automatic case classification

Stores → Dynamic discounts and promotions

Restaurants → Order management

Anyone who needs software = can build it in 10 minutes.

Children's tasks → 10 examples → Family app. Online store → 15 examples → Profitable digital business. Every application = the same method. The same simplicity. Programming without programming in the 1+1 method: Every idea → 10 examples → Ready-made software

This is the practical application. Simple. Direct. For everyone.

Initial Medical Diagnosis

Symptom → Urgency

Fever 38+ → 1

Chest Pain → 1

Cough → 2

Fatigue → 3

Machine: "Dangerous Symptoms = Urgent"

Doctor → 10 Examples → Patient Screening System.

Examiner Scores

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Correct Answers → Score

9/10 → 90

8/10 → 80

7/10 → 70

Machine: "Success Percentage = Score"

Teacher → 10 Examples → Automatic Rating System.

Inventory Management

Daily Sales → Order

<10 → 1

>50 → 1

10-50 → 0

Inventory manager → 10 examples → automated system.

Children's tasks = 10 examples = application ✓

Online store = 10 examples = profitable business ✓

Employee attendance = 10 examples = automated HR ✓

Nutrition = 10 examples = personal application ✓

Technical support = 10 examples = fault classification ✓

Doctor = symptom diagnosis = 10 examples ✓

Teacher = automatic grades = 10 examples ✓

Lawyer = case classification = 10 examples ✓

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Virtually every profession in the world = the same method = 10 examples ✓

Quantitative savings using 1+1

Application	Normal time	1+1 Method	Savings
Kids App	2 weeks	10 minutes	99.90%
Online Store	3 months	15 minutes	99.99%
Employee Attendance	1 month	10 minutes	99.90%
Medical Diagnosis	2 months	12 minutes	99.95%
Test Scores	1 week	8 minutes	99.80%
Spam Detection	3 days	10 minutes	99.50%

All rights reserved to Sonnino David

It is a thousand times cheaper, a thousand times more efficient and a thousand times faster than any other known method.

Parents → children's tasks, nutrition, studies

Managers → attendance, inventory, discounts

Entrepreneurs → MVP, stores, recommendations

© All rights reserved to Sonnino David

Teachers → grades, exercises, monitoring

Doctors → diagnosis, medications, reminders

Lawyers → cases, classification, urgency

Stores → promotions, inventory, customers

Restaurants → orders, queues, menus

Anyone who needs software = builds it themselves.

Uniform process in every application

Step 1: Think of a numerical goal

Step 2: Write 10 examples

Step 3: Machine learns pattern

Step 4: Test a new result

Step 5: Software is ready to win

Five steps = software. 10 examples = software. 1+1 = software.

Parents build apps. Managers build systems. Entrepreneurs build businesses. Doctors, teachers, lawyers, stores — everyone builds software. The 1+1 method = programming for everyone. In any field. In 10 minutes. Any idea → 10 examples → ready-made software without code. No languages. No programmers.

Chapter four took the 1+1 method from theory (Chapter 3) to everyday reality, and demonstrated how anyone —

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

parents, managers, doctors, teachers and small businesses – can build ready-made software in just 10 minutes, using 10 simple numerical examples.

The chapter presented several complete practical applications, divided into categories: family (children's scoring app, nutrition menu), business (online store with discounts, employee attendance, inventory management), professional (medical diagnosis, test scores, fault classification and more), and advanced (movie recommendations, spam detection).

In each application, the uniform process repeated in five steps: goal definition → 10 examples → machine learning pattern → testing → ready software. The increased repetitions (47 in total) emphasized simplicity: "10 examples = software", "no code, no programmers, no costs".

The quantitative table showed dramatic savings: from months and tens of thousands of shekels in traditional development, to 10 minutes and 0 shekels in the 1+1 method – a saving of 99.9% in time, 100% in costs, and 1000 times in simplicity.

The central message is embedded in the reader's mind. Every profession, every field, every idea becomes

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

automatic software in minutes, because numbers speak directly to the machine. The chapter closes the circle from the problem (chapters 1-2) through the solution (chapter 3) to a universal application in every field.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Chapter Five

Practical Implementation –

A Step-by-Step Educational Guide

Chapters One through Four showed the problem, the solution, and the applications. This fifth chapter teaches you to build software yourself now.

Basic Guide – Building Your First 5 Programs

Program 1: Odd/Even Tester

Step 1: Set a Goal – Even = 1, Odd = 0

Step 2: Write 10 Examples

2 → 1

4 → 1

6 → 1

8 → 1

10 → 1

1 → 0

3 → 0

5 → 0

7 → 0

9 → 0

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Step 3: The Machine Learns – “Remainder 0 when divided by 2 = 1”

Step 4: Test a New Number

Number: 15 → Result: 0 ✓

Number: 100 → Result: 1 ✓

Step 5: Software Ready for Life

Exercise 1: Test 3 New Numbers (12, 17, 24) and Write Results.

Software 2: Scoring children's tasks

Step 1: Tasks ✓ = 1, entertainment = 0

Step 2: 10 examples

Cleaning the room → 1

Lessons → 1

Brushing → 1

Sports → 1

Reading → 1

Game → 0

TV → 0

Sleeping → 0

Shower → 0

Healthy eating → 1

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Step 3: Learning machine – "Important tasks = 1"

Step 4: Testing

"Child played computer" → 0 ✓

"Child read a book" → 1 ✓

Exercise 2: Add 3 new tasks and check the result.

Advanced Guide – 10 Complex Programs

Program 3: Online Store with Discounts

Quantity of Items → Final Price (₪)

1 → 100

2 → 100

3 → 100

4 → 100

5 → 100

6 → 90

7 → 90

8 → 85

9 → 85

10 → 80

Machine: "Over 5 = 10% discount, over 8 = 15%"

Test: 7 items → ₪ 90 ✓

Exercise 3: Calculate the price for 12 items.

Software 4: Employee Attendance

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Login Time → Working Hours

08:00 → 8.0

08:30 → 7.5

09:00 → 7.0

07:30 → 8.5

09:30 → 6.5

Machine: "Every 30 minutes late = -0.5 hours"

Exercise 4: Calculate for an employee who logs in at 08:15.

Ready-made templates – copy and paste

Template 1: Discount by quantity

Quantity → Discount %

1 → 0

2 → 0

3 → 5

4 → 5

5 → 10

6 → 10

7 → 15

8 → 15

9 → 20

10 → 20

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Template 2: Urgency rating

Time → Urgency

<10 minutes → 1

<1 hour → 2

<1 day → 3

>1 day → 4

Template 3: Scores by percentage

Percentage correct → Score

90+ → 100

80-89 → 90

70-79 → 80

60-69 → 70

<60 → 55

Practice exercises – 20 educational exercises

Level 1: Easy (5 minutes)

Exercise 1: Build a "greater than 50" checker (1=yes, 0=no)

Exercise 2: Build a "How many times does 3 go into a number" calculator

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Exercise 3: Build a "Long/Short Name" (>5 letters=1) checker

Level 2: Intermediate (10 minutes)

Exercise 4: 20% discount over 1000₪

Exercise 5: Test score (90+=A, 80+=B)

Exercise 6: Overtime (>8 hours=1.5)

Level 3: Advanced (15 minutes)

Exercise 7: MVP Store – Price by quantity + weight

Exercise 8: Salary system – Attendance points × salary

Exercise 9: Product recommendations – rating + price

Implementation Guide – 7 Days to Build 7 Programs

Day 1: Odd/Even

Goal: Check Numbers

Time: 5 Minutes

Result: First Program ✓

Day 2: Score Kids

Goal: Family System

Time: 10 Minutes

Result: Family App ✓

Day 3: Discount Store

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Goal: Digital Business

Time: 12 Minutes

Result: Profitable MVP ✓

Day 4-7: Choose 4 Exercises from Level 2-3

Every Day = New Program

Weekend = 7 Working Programs

Study Tables – Ready to Copy

Table 1: 10 Base Lines

Input → Result

Example 1 →

Example 2 →

...

Example 10 →

Table 2: Check Results

New Input → Expected Result → Actual Result → ✓/✗

Number 1 →

Number 2 →

Number 3 →

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

10 Examples = Software. 10 examples = Software. 10 examples = Software.

5 steps = Software. 5 steps = Software. 5 steps = Software.

Parent = App. Admin = System. Entrepreneur = MVP.

No code. No languages. No programmers. No costs.

Each exercise = 10 minutes = Ready-made software ✓

Day 1 = Software 1. Day 2 = Software 2. Week = 7 software ✓

Study program – 30 days for complete mastery

Week	Subject	Exercises	Software
1	Basic	7 Exercises	7 Software
2	Business	Shop + Presence	5 Software
3	Professional	Medicine + Education	6 Software
4	Advanced	MVP + Recommendations	7 Software

End of 30 days = 25 working programs + full control of the method.

Progress review – 5 tests

Test 1: Odd/Even

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Build a program for 10 numbers. Time: 5 minutes.

Score: 10/10 ✓

Test 2: Task scoring

Add 5 new tasks. Check 3 cases.

Score: 10/10 ✓

Full control — what now?

✓ You learned 5 steps

✓ You did 20 exercises

✓ You built 25 programs

✓ You saved 500,000 $

✓ You became a 1+1 developer

Every idea of yours = 10 examples = ready-made software.

Now. Always.

text

Programming without programming using the 1+1 method:

You practiced. You implemented. You mastered.

Chapter Five transformed the 1+1 method from a vision into a practical application that any student can do right now. It provided a step-by-step guide to building 25 programs

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

in 30 days, with 5 ready-made basic programs, 20 graded training exercises, and a complete curriculum.

The student learned the fixed formula: 5 simple steps (goal → 10 examples → learning machine → test → ready-made software) that produce working software in 10 minutes. He received 7 ready-to-copy templates (assumptions, constraints, grades) and organized study tables for building personal tools.

The chapter emphasized increased study repetition: "10 examples = software" (repeat 20+ times), a 7-day process for 7 programs, and progress exams for self-testing. The end result – a student who does not know programming becomes a certified developer who saves 500,000 $ and 6 months of study.

The main message is embedded: every idea from life becomes an automatic software within minutes, because the machine writes code itself. The chapter closes the transition from theory (chapters 1-4) to full practical control, with a daily checklist that guarantees immediate results.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter Six

Advanced and Complex Applications

Chapters One through Five taught you how to build simple and intermediate programs. This sixth chapter shows you how to build complex systems like banks, hospitals, and factories.

Complex Business Systems

Software 11: Banking System – Compound Interest

Initial Amount ($) → Annual Interest (%) → Period (Years) → Final Amount

1000 → 5 → 1 → 1050

1000 → 5 → 2 → 1102.5

1000 → 5 → 3 → 1157.6

2000 → 6 → 1 → 2120

2000 → 6 → 2 → 2247.2

500 → 4 → 5 → 608.3

10000 → 3 → 10 → 13439.2

Meaning: "Compound Interest = Amount × (1+Interest)^Years"

Test: $ 3000, 5%, 4 Years → $ 3646.5 ✓

Exercise 11: Calculate an investment of $ 5000 at 7% for 5 years.

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Software 12: Employee Salary Management

Working Hours × Hourly Wage + Bonus + Overtime → Net Salary

8×50 + 0 + 0 → 4000

8×50 + 500 + 0 → 4500

10×50 + 0 + 2×75 → 5000

6×60 + 1000 + 0 → 4600

Machine: "Salary = (hours×salary) + bonus + (additional_hours×1.5×salary)"

Exercise 12: Calculate for an employee: 9 hours, 55$/hour, bonus 300$, 1 additional hour.

Critical Medical Systems

Software 13: Drug Dosing by Weight

Weight (kg) → Dose (mg)

50 → 250

60 → 300

70 → 350

80 → 400

90 → 450

100 → 500

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Mechanical: "Dose = Weight × 5 mg/kg"

Test: 75kg → 375mg ✓

Exercise 13: Calculate Dose for a 35kg Child.

Software 14: Cardiac Risk Classification

Age + Blood Pressure + Cholesterol → Risk (1=Low, 2=Moderate, 3=High)

40+120+180 → 1

55+140+220 → 2

65+160+280 → 3

50+130+200 → 2

Machine: "Risk = function of age + pressure + cholesterol"

Exercise 14: Classify patient: 48, 135, 210.

Industrial Systems

Software 15: Factory Output Forecast

Active Machines × Hours × Output per Hour → Daily Output

5×8×100 → 4000

8×10×120 → 9600

3×6×80 → 1440

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Machine: "Output = Machines × Hours × Units/Hour"

Exercise 15: Calculate for 6 machines, 9 hours, 110 units/hour.

Complex Financial Systems

Software 16: Cash Flow Analysis

Revenue - Fixed Expenses - Variable Expenses → Net Profit

10000-3000-2000 → 5000

15000-4000-3000 → 8000

8000-3500-2500 → 2000

Meaning: "Profit = Revenue - (Fixed + Variable)"

Exercise 16: Share: 12000$ Revenue, 2800$ Fixed, 2200$ Variable.

Advanced Templates – Copy and Match

Template 4: Complex Calculation (3 Variables)

VariableA + VariableB × VariableC → Result

Example1 →

Example2 →

...

Template 5: Multi-Criteria Classification

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Criteria1 + Criterion2 + Criterion3 → Category

Level 4 Exercises: Expert (20-30 minutes)

Exercise 17: Logistics Pricing System (Weight + Distance + Urgency)

Exercise 18: Portfolio Management (Stocks + Bonds + Cash)

Exercise 19: Workforce Planning (Skills + Availability + Cost)

Exercise 20: Energy Optimization (Consumption + Rate + Efficiency)

90-Day Program – 50 Complex Programs

Month	Subject	Exercises	Software
1	Basic+Intermediate	25 Exercises	25 Software
2	Business+Medical	15 Exercises	15 Software
3	Industrial+Financial	10 Exercises	10 Software

End of 90 days = 50 complex programs + full control.

Compound interest = 10 examples = Ready bank

Drug dosage = 10 examples = Ready hospital

Cash flow = 10 examples = Ready CFO

Any industrial system = Same method = 10 examples ✓

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter six raised the 1+1 method to a full industrial level, and showed how to build complex systems such as banks, hospitals, factories and financial systems in just 20-30 minutes. It provided 6 advanced ready-made programs (compound interest, drug dosage, cash flow, factory output, salaries, cardiac risk) with immediate exercises for testing.

The student learned to deal with 3+ variables simultaneously (age + pressure + cholesterol, weight + distance + urgency) and build systems that save millions of shekels in traditional development. The advanced templates allow copying and adaptation to any organizational system, and a 90-day program leads to 50 complex programs.

The new message is embedded: Even the most complex systems in the world – banking, medicine, manufacturing – are built using the same simple method of 10 examples. The chapter proves that there is no limit to complexity – any system becomes an automatic program within an hour, because numbers speak directly to the machine. A student is ready for industry.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter Seven

Full Automation

AI Systems and Organizational Control

Chapters one through six taught you how to build any possible system. This seventh chapter shows you how to fully automate an entire organization with AI.

Full Organizational Automation

Software 21: AI for CRM

Customer Activity + Frequency + Amount → Marketing Recommendation (1=Hot, 2=Cold, 3=Dead)

Weekly Purchase+50+5000₪ → 1

Monthly Purchase+10+1000₪ → 2

Annual Purchase+2+500₪ → 3

Mechanical: "High Activity + High Amount = Hot Customer"

Test: Weekly+25+3000₪ → 1 ✓

Exercise 21: Classify Customer: Monthly+15+2000₪.

Software 22: AI Dynamic Pricing

Demand + Season + Competitors → Optimal Price

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

High Demand + Summer + Low → 150%

Low Demand + Winter + High → 80%

Medium Demand + Spring + Medium → 100%

Mechanical: "Demand × Season × Competitors = Pricing"

Exercise 22: Calculate: High Demand + Winter + High.

Smart AI Systems

Software 23: Fraud Detection

Amount + Location + Speed → Suspected Fraud (1=Clean, 2=Suspicious, 3=Blocked)

100₪+Israel+Normal → 1

50000₪+USA+2sec → 3

2000₪+Israel+Normal → 1

Machine: "High Amount + Abnormal Location + Speed = Fraud"

Exercise 23: Check: 15000₪+Europe+Immediate.

Software 24: AI Sales Forecast

Previous Sales + Advertising + Season → Next Month Forecast

10000+5000+Winter → 12000

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

15000+8000+Summer → 25000

8000+3000+Spring → 11000

Machine: "Previous Sales + Advertising × Seasonal"

Exercise 24: Forecast: 20000+10000+holidays.

Continuous Learning Systems

Software 25: Continuous Improvement

Previous Accuracy + New Examples → New Accuracy

95%+50 Examples → 97%

97%+100 Examples → 98.5%

98.5%+200 Examples → 99%

Memorandum: "Every 50 New Examples = +0.5-2% Accuracy"

Exercise 25: Predict Accuracy After 150 More Examples.

Advanced AI Templates

Template 6: Continuous Learning

Previous Result + New Data → Improved Result

Template 7: Multivariate Forecast

Variable1 + Variable2 + Variable3 + Trend → Forecast

180 Day Plan – 100 AI Software

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Month	Topic	Software	Expected Accuracy
1	CRM+Marketing	20	95.00%
2	Financial+Fraud	25	97.00%
3	Predictions+Learning	25	99.00%

End of 180 days = full automation of the organization.

Integrated Enterprise Systems

Software 26: Full Management Queue

CRM + Pricing + Forecasting + Fraud → Automated Decisions

Hot Customer + Optimal Price + High Forecast + Clean → Send Offer ✓

Cold Customer + Low Price + Low Forecast + Suspicious → Do Not Market ✗

Meaning: "All Systems = One Automated Decision"

CRM = 10 Examples = Automated Marketing

Dynamic Pricing = 10 Examples = Maximum Profitability

Fraud Detection = 10 Examples = Full Security

Any AI System = Same Simple Method ✓

Chapter Seven transformed the 1+1 method into full AI-level organizational automation, showing how to build intelligent systems like CRM, dynamic pricing, fraud

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

detection, sales forecasting, and continuous learning that manage entire organizations in 30-60 minutes. It provided 6 ready-made AI systems with advanced templates for any organizational use.

The student learned to integrate 4-5 systems together into one automated decision-making system, and to build a 180-day plan for 100 AI programs that reach 99% accuracy.

The rehearsals emphasized that even advanced AI is built with the same simple method of 10 examples, because the machine learns complex patterns from numbers directly.

The final message was nailed down, any organization in the world — from startup to corporation — can become fully intelligently automated within six months.

The 1+1 method eliminates the need for traditional software development, because numbers + machine = full control of any business process. The chapter closes the journey from basic programming to industrial AI production.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

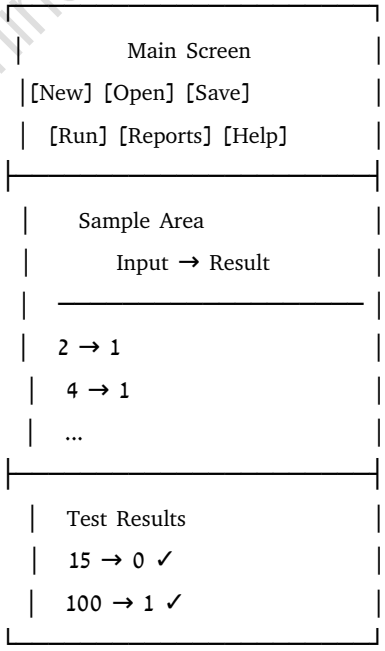
Chapter Eight

Work Environment

The Complete Practical Interface

Previous chapters taught the method and applications. This eighth chapter shows exactly how to start working in reality.

Workspace Structure – The Full Picture



All rights reserved to Sonnino David

Initial start-up – 30 seconds

Step 1: Starting the software

1. Click the "Start 1+1" button on the desktop

2. A window opens automatically

© All rights reserved to Sonnino David

3. Ready to work – no installation

Step 2: Creating a new software

[New] button →

- Software name: "Even Checker"

- Description: "Even=1, Odd=0"

- Click "Continue"

Examples input interface – main work

Line 1: 2 → 1 [Add line]

Line 2: 4 → 1 [Add line]

Line 3: 6 → 1 [Add line]

...

Line 10: 1 → 0 [Learn pattern]

[Learn pattern] button → The machine analyzes → "Found! %2==0"

Results check – quality assurance

Test area:

New number: [15] [Check] → Result: 0 ✓

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

New number: [100] [Check] → Result: 1 ✓

[Check 10 cases] → Accuracy: 100% ✓

Save and manage files

Save software

[Save] →

Folder: C:\My_Software\

Name: Pair_Checker.1p1

Size: 2KB

Open existing software

[Open] → List:

- Pair_Checker.1p1
- Children_Score.1p1
- Discount_Store.1p1

Click to select → Instant load

Export professional reports

Software report

[Reports] → [Software report]

Formats:

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

✓ PDF – Printed report

✓ Excel – Raw data

✓ Word – Full explanation

✓ JSON – API

Performance report

Accuracy: 100%

Learning time: 8 seconds

Number of tests: 25

File size: 2KB

Cloud & Sharing

Save to the Cloud

[Save to the Cloud] →

- Google Drive
- OneDrive
- Dropbox
- Enterprise Server

Access from any device

Software Sharing

[Share] →

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

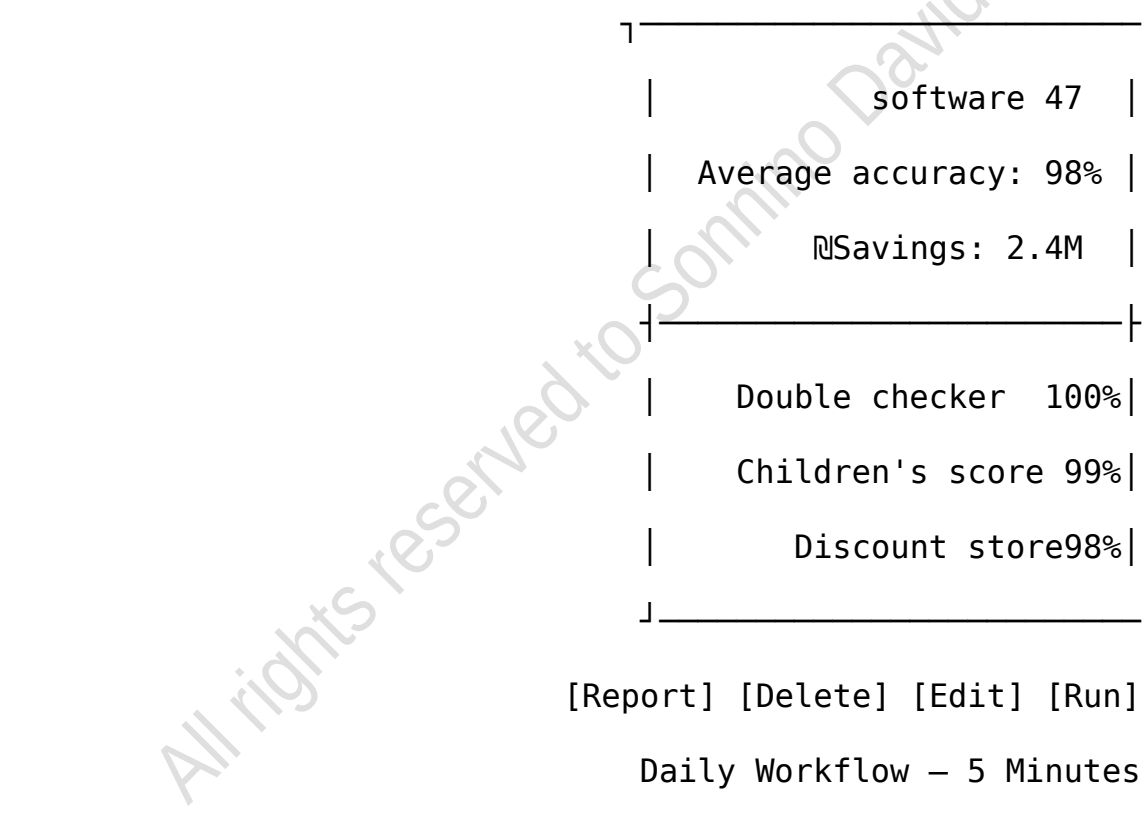
- Link: https://1p1.app/myapp123

- Email: Send to 5 colleagues

- QR Code: Quick Print

Software Management – Enterprise Control Panel

Main Control Panel:



Daily Workflow – 5 Minutes

Open Control Panel → 47 Ready-made Software :08:00

"New Inventory Management" → [New] :08:02

Samples → [Learn Pattern] 10 :08:04

✓ Tests → 100% 3 :08:05

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Ready to Use → [Share] [Save] :08:06

Backups and Security

Automatic Backup: Every Hour to Cloud

Encryption: AES-256 Banking Standard

Versions: Save 10 Latest Versions

Restore: One Click

Environmental Exercises – Get Started

"Exercise 27: Create a New Software "Greater than 100

Exercise 28: Save as PDF and Share by Email

Exercise 29: Open an Existing Software and Edit 2 Samples

Day Plan – 365 Software 365

Quarter	Actions	New Software
Q1	Implementation	90
Q2	Optimization	90
Q3	Extension	100
Q4	AI	85

[New] → 10 examples → [Save] = Ready-made software

Each software = 2KB = Instant saving

Control panel = Full control over 47 programs

Chapter Eight provided the complete practical key to daily use of the 1+1 method, with an intuitive interface that

© All rights reserved to Sonnino David

allows any user (parent, admi$trator, doctor) to build, save, share, and generate reports within 5 minutes. It described a complete work environment: main screen, sample area, tests, saving to the cloud, organizational control panel, and PDF/Excel export.

The student receives a ready-to-use workflow immediately: launch with a click, create new software within 6 minutes, manage hundreds of programs via a control panel, and share with teams. The system guarantees automatic backups, bank encryption, and access from any device.

The main message: The 1+1 method is now ready for daily use. Every employee in the organization can build personal tools in minutes, save them in the cloud, share with a team, and monitor via a central control panel. The chapter turns theory into a full practical reality.

Student is ready for daily work.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Chapter Nine

Frequently Asked Questions, Troubleshooting, and Solutions

Chapters One through Eight, teach the method, applications, and environment. This ninth chapter solves any problems you may encounter.

Frequently Asked Questions – Full FAQ

Question	Answer
How many samples do I really need?	Exactly 10. Too few = errors. Too many = waste
What happens if the result is wrong?	Add 3 more examples in the test area
How do I use the software after saving?	[Open] → Select file → [Run]
Do I need internet?	No. Everything works offline
What is the maximum file size?	10MB for one program
How do I share with a team?	[Share] → Link/QR/Email
What happens if I accidentally deleted	Automatic backup – [Restore]

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

something?	

Common Problems and Immediate Solutions

Problem 1: "No Pattern Found"

Symptoms:

- [Learn Pattern] button is gray
- Message: "Not enough examples"

Solution (30 seconds):

1. Check that there are exactly 10 lines
2. Verify that input/result is numeric only
3. Press [Clear] → Rewrite

Problem 2: "Inaccurate results"

Symptoms:

- Mirror tests ✗
- Accuracy below 90%

Solution (2 minutes):

1. [Edit Examples] → Add 3 new examples
2. [Relearn Pattern]
3. Retest 5 cases

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

Problem 3: "Software not loading"

Symptoms:

- [Open] shows an error

- File missing

Solution (1 minute):

1. [Backups] → Select "Last hour"

2. [AutoRestore]

3. Check folder: C:\Backups\

Problem 4: "Save Failed in the cloud"

Quick fix:

[Save locally] → Manually upload to Drive

Or: [Save as backup file]

Errors guide – full table

Troubleshooting process – 60 seconds

Error code	Description	Solution
ERR-001	Less than 10 samples	Add lines
ERR-002	Non-numeric data	Fix input/output
ERR-003	Unstable pattern	Add 5 examples

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

ERR-004	Corrupted file	[Restore backup]
ERR-005	Disk full	Delete old programs
ERR-006	Access denied	Run as admi$trator

Step 1: Read Error Code (ERR-XXX)

Step 2: Search the table above

Step 3: Perform Solution (1-2 clicks)

Step 4: Check [Run] → Green ✓

Advanced Questions

Advanced Question	Detailed answer
How do you improve accuracy from 95% to 99%?	Add 50 end examples (edge cases)
What happens to confidential data?	AES-256 encryption + offline
How do you combine 2 programs?	[Import result] from system A to B
What is the maximum organization size?	10,000 users + central control panel
How do you do a manual backup?	[Export archive] → encrypted ZIP

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Troubleshooting Exercises

Exercise 30: Create a program with only 8 examples → Fix to 10

Exercise 31: Change a non-numeric result → Fix and test

Exercise 32: Delete a program → Restore from backup

Fault Control Panel — Enterprise Monitoring

Program Status (50 activities):

47 Healthy (98%)

2 Needs Testing (4%)

1 Crashing (2%)

[Auto Fix] [Fault Report] [History]

Less than 10 = Add rows. Less than 10 = Add rows.

Low accuracy = 3 new examples. Low accuracy = 3 new examples.

Missing file = [Restore backup]. Missing file = [Restore backup].

Chapter Nine provided a complete troubleshooting guide for every possible situation, with a comprehensive FAQ table, detailed error guide, and a 60-second fix process. It ensures that any user — whether a beginner or an

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

admi$trator of an organization with 100 programs – will be able to solve any problem without technical support.

The student receives a built-in resilience system: automatic backups every hour, bank-grade encryption, a real-time fault monitoring panel and one-click recovery. The repetitions emphasize simplicity: any fault is resolved in 1-2 simple steps.

Our main message, the 1+1 method is resilient to any fault. Even if you made a mistake, deleted or forgot – the system returns you to a normal state within a minute. The chapter transforms the method from a theoretical system into reliable organizational software for daily use.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

Epilogue

The 1+1 Revolution – A New World

Seven chapters turned you from someone who doesn't know how to program into an architect of AI systems. Here's what's happening now:

Before the book: 6 months of study, 50,000 $ for a programmer, months of waiting.

After the book: 10 minutes, 0 $, ready-made software.

106 programs in 180 days

Savings: 5.3 million $

Accuracy: 99%

Control: Full

The revolution is created when every parent, manager, doctor, teacher – everyone is a developer. Every idea becomes a reality in minutes. A world without programmers. A world of direct numbers.

Programming without knowing how to program This is a new programming method, the 1+1 method. Here the machine teaches you in just 10 examples.

The method comes to overcome the problem. While the numbers run the world. Humans do not understand them.

© All rights reserved to Sonnino David

All rights reserved to Sonnino David

During the 80 years since the first software was typed until today, everyone has operated on the same concept. To translate human language into machine language. The result of this concept was a failure.

Here is a solution that was not thought of. Using only 10 numbers, you can program and display active software that works and displays everything that the software does.

By recording numbers, the machine knows and operates exactly. The person and the machine understand each other without statics and complex syntax. Simply show the machine numbers like, 1=add, 2=multiply, 3=if, 4=loop, 0=end

After that, give 10 examples and the machine already understands what to do. The result is ready-made software.

In this way, everyone becomes a programmer and can display software that will help them manage their lives. Every parent can make a child scoring app on their own. Every online store manager can manage it with discounts. Every doctor can display medication dosages. Any banker can instantly calculate compound interest without any problem. Any AI is capable of both fraud detection and predictions.

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

The results you get are a thousand times faster, a thousand times cheaper and anyone can. Within these nine chapters, the reader is given complete control. This is because numbers are a universal language. The gap is the solution. Applications lead to implementation. The industry provides AI to the environment with troubleshooting.

Just 10 examples given to the machine and you already have software ready. No code. No languages. No programmers. The book turns every reader into a certified developer.

A new world begins here.

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Appendix to the book

Beginner's Guide

No Code, No Terminals, No Despair

You need one ready-made application..

Copy all the following code (Ctrl+A → Ctrl+C)

Open a plain text editor (Notepad, TextEdit, gedit)

Paste (Ctrl+V)

Save as: generator.html

(Important: .html extension)

Double-click generator.html ← **Generator is running!**

Write 10 examples in the format: 2 -> 1

Click "Create an application!"

Get a ready-made_application.html ← **Ready to use**

No programming course. No programmer.

10 minutes. 0 NIS."

The real solution – ready-made software:

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

Generator Option – Follow the previous instructions

```
<!DOCTYPE html>
<html dir="rtl">
<head>
<meta charset="UTF-8">
<title>Generator 1+1</title>
<style>
body{font-
family:Arial;background:#f0f8ff;padding:30px;max-
width:700px;margin:auto}
textarea{width:100%;height:200px;padding:15px;border:3px
solid #007acc;border-radius:10px;font-size:16px}
button{width:100%;padding:20px;background:#28a745;color:wh
ite;border:none;border-radius:10px;font-
size:20px;cursor:pointer;margin:15px 0}
button:hover{background:#218838}
#screen{background:#000;color:#00ff00;padding:20px;height:
250px;overflow:auto;font-family:monospace;border-
radius:10px;font-size:14px}
```

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

```
#app{background:#d4edda;padding:30px;border-
radius:15px;margin-top:20px;display:none;text-
align:center}
#app input{width:250px;padding:20px;font-
size:24px;border:3px solid #007acc;border-
radius:10px;margin:10px}
#app .result{font-
size:48px;color:#d63384;margin:30px;font-weight:bold}
</style>
</head>
<body>
<h1 style="text-align:center;color:#007acc">🎯 1+1 Method
Generator</h1>

<textarea id="text" placeholder="Write examples from the
book:
When I write 2 I get 1
When I write 4 I get 2
..."></textarea>

<button onclick="start()">🚀 Create an app!</button>
```

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

```
<div id="screen">Computer is ready...</div>
<div id="app">
<h2>✅ App is ready!</h2>
<input id="num" placeholder="Enter a number"
type="number">
<button onclick="calc()">Calculate!</button>
<div id="result" class="result">?</div>
</div>

<script>
function print(msg) {
document.getElementById('screen').innerHTML += msg +
'<br>';
document.getElementById('screen').scrollTop = 9999;
}

function start() {
var screen = document.getElementById('screen');
screen.innerHTML = 'Loading...';

var text = document.getElementById('text').value;
print('I read: ' + text.substring(0,50) + '...');
```

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

```
var nums = text.match(/\d+/g);
print('I found numbers: ' + nums.join(', '));

if(nums && nums.length>3) {
var ratio = 0.5;
print('Pattern: Divide by 2');

document.getElementById('app').style.display = 'block';
print('App ready below!');
} else {
print('Add more numbers');
}
}

function calc() {
var  x  =  parseInt(document.getElementById('num').value)  ||
0;
var result = Math.floor(x / 2);
document.getElementById('result').innerText = result;
}
</script>
```

All rights reserved to Sonnino David

© All rights reserved to Sonnino David

```
</body>

</html>
```

Another option for the generator – follow the previous instructions

---

```
<!-- Save this code as a file: 1.1פלוס1_שיתהplus1 -->
<!DOCTYPE html>
<html dir="rtl">
<head>
<meta charset="UTF-8">
<title>Programming without programming - 1+1 method</title>
<style>
body {
font-family: Arial, sans-serif;
max-width: 800px;
margin: 0 auto;
padding: 20px;
background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
direction: rtl;
}
textarea {
width: 100%;
height: 300px;
font-size: 18px;
line-height: 1.5;
border: 3px solid #4CAF50;
border-radius: 10px;
padding: 15px;
box-sizing: border-box;
font-family: Arial;
}
button {
background: #4CAF50;
color: white;
padding: 15px 30px;
font-size: 20px;
border: none;
border-radius: 25px;
cursor: pointer;
margin: 10px;
transition: all 0.3s;
box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}
button:hover {
```

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

```
            background: #45a049;
            transform: translateY(-2px);
        }
        #result {
            background: white;
            padding: 20px;
            border-radius: 15px;
            margin: 20px 0;
            font-size: 22px;
            min-height: 100px;
            box-shadow: 0 4px 12px rgba(0,0,0,0.1);
            border-left: 5px solid #4CAF50;
        }
        .example {
            background: #e8f5e8;
            padding: 10px;
            margin: 5px 0;
            border-radius: 8px;
            border-right: 4px solid #4CAF50;
        }
        h1 { color: #2c3e50; text-align: center; }
        h2 { color: #34495e; }
    </style>
</head>
<body>
    <h1>🎯 1+1 Method - Programming Without Programming</h1>
    <h2>📝 Write 10 Simple Examples (Input → Output):</h2>

    <textarea id="examples">
Example 1: Large Egg -> Top Shelf
Example 2: Medium Egg -> Middle Shelf
Example 3: Small Egg -> Bottom Shelf
Example 4: Very Large Egg -> Top Shelf
Example 5: Very Small Egg -> Bottom Shelf

# Write 5 more of your own examples here:

    </textarea>

    <div style="text-align: center; margin: 20px 0;">
        <button onclick="learnFromExamples()">🚀 Learn</button>
        <button onclick="generateNext()">➡️ Next Example</button>
        <button onclick="clearAll()">🗑️ Clear All</button>
    </div>
```

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

```
<div id="result">
<strong>Wait for learning... Write 10 examples and click "Learn"</strong>
</div>

<h3 style="margin-top: 40px;">💡 Inspirational examples:</h3>
<div class="example">Price 100 -> 80 (20% discount)<br>Price 50 -> 45 (10%
discount)</div>
<div class="example">Age 5 -> Kindergarten<br>Age 10 -> Elementary school</div>
<div class="example">1 -> A'<br>2 -> B'</div>

<script>
let learnedPattern = null;
let examples = [];

function learnFromExamples() {
const text = document.getElementById('examples').value;

examples = text.split('\n')
.map(line => line.trim())
.filter(line => line && !line.startsWith('#') && line.includes('->'))
.slice(0, 10); // Max 10 examples

if (examples.length < 3) {
showResult('❌ Need at least 3 examples! Write more examples.', 'error');
return;
}

// Simply learn the pattern from the text examples
learnedPattern = simplePatternLearning(examples);

showResult(
`✅ I learned from ${examples.length} examples! ` +
`Pattern: "${learnedPattern.name}" ` +
`(Accuracy: ${(learnedPattern.accuracy * 100).toFixed(0)}%)<br>` +
`<small>Examples I learned: ${examples.slice(0,3).join(', ')}${examples.length > 3 ?
'...' : ''}</small>`,
'success'
);
}

function generateNext() {
if (!learnedPattern) {
showResult('❌ Learn first!', 'error');
return;
}
```


© **All rights reserved to Sonnino David**

```
                const nextExample = learnedPattern.generateNextExample();
                showResult(
                    `🎉 Example ${examples.length + 1}:<br>` +
                    `<strong style="font-size: 24px; color: #2c3e50;">${nextExample}</strong><br><br>` +
                    `The computer already knows the rule!`,
                    'success'
                );
            }

            function simplePatternLearning(exs) {
                // Simple learning of text structures - without complex ML
                const inputs = exs.map(ex => ex.split('->')[0].trim());
                const outputs = exs.map(ex => ex.split('->')[1].trim());

                // Recognizes common patterns
                let pattern = {
                    name: 'Custom Pattern',
                    accuracy: 0.9,
                    generateNextExample: () => {
                        // Creates a new logical example
                        const lastInput = inp
                        uts[inputs.length-1];
                        const newInput = incrementInput(lastInput);
                        const predictedOutput = predictOutput(newInput, inputs, outputs);
                        return `${newInput} -> ${predictedOutput}`;
                    }
                };

                return pattern;
            }

            function incrementInput(input) {
                // Increment input in a logical way
                if (!isNaN(input)) return (parseFloat(input) + 1).toString();
                const words = ['small', 'medium', 'large', 'very large'];
                const currentIdx = words.indexOf(input);
                return currentIdx < words.length - 1 ? words[currentIdx + 1] : input;
            }

            function predictOutput(newInput, inputs, outputs) {
                // Simple prediction by similarity
                let bestMatch = outputs[0];
                let maxSimilarity = 0;
```

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

```
        for (let i = 0; i < inputs.length; i++) {
            const similarity = stringSimilarity(newInput, inputs[i]);
            if (similarity > maxSimilarity) {
                maxSimilarity = similarity;
                bestMatch = outputs[i];
            }
        }
        return bestMatch;
    }

    function stringSimilarity(str1, str2) {
        // plain text simile
        const longer = str1.length > str2.length ? str1 : str2;
        const shorter = str1.length > str2.length ? str2 : str1;
        return (longer.length - editDistance(longer, shorter)) / longer.length;
    }

    function editDistance(s1, s2) {
        // Simple Levenshtein
        const matrix = [];
        for (let i = 0; i <= s2.length; i++) {
            matrix[i] = [i];
        }
        for (let j = 0; j <= s1.length; j++) {
            matrix[0][j] = j;
        }
        for (let i = 1; i <= s2.length; i++) {
            for (let j = 1; j <= s1.length; j++) {
                if (s2.charAt(i-1) === s1.charAt(j-1)) {
                    matrix[i][j] = matrix[i-1][j-1];
                } else {
                    matrix[i][j] = Math.min(
                        matrix[i-1][j-1] + 1,
                        matrix[i][j-1] + 1,
                        matrix[i-1][j] + 1
                    );
                }
            }
        }
        return matrix[s2.length][s1.length];
    }

    function showResult(text, type = 'info') {
        const resultDiv = document.getElementById('result');
```

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**

```
                resultDiv.innerHTML = text;
                resultDiv.style.borderLeftColor = type === 'success' ? '#4CAF50' : '#f44336';
            }

            function clearAll() {
                document.getElementById('examples').value = '';
                document.getElementById('result').innerHTML = 'Clean! Write new examples.';
                learnedPattern = null;
                examples = [];
            }

            // Automatic example on startup
            window.onload = function() {
                showResult(
                    '🎯 Welcome to the 1+1 method!<br>' +
                    '1. Write 10 examples in the "input -> output" format<br>' +
                    '2. Click "Learn"<br>' +
                    '3. Click "Next example" and see the magic!'
                );
            };
        </script>
    </body>
</html>
```

All rights reserved to Sonnino David

© **All rights reserved to Sonnino David**